Програма користувача може програмувати байт за допомогою запису лог. 1 в розряд EEWE лише впродовж цих чотирьох тактів системної синхронізації. Якщо буде встановлений розряд EEWE, але без установки розряду EEMWE, то процес програмування розпочатий не буде. Фрагмент перевірки правильності процедури запису наведено з використанням інструментального засобу програмування МК фірми Atmel – AVR Studio 4.0 (рис. 2).

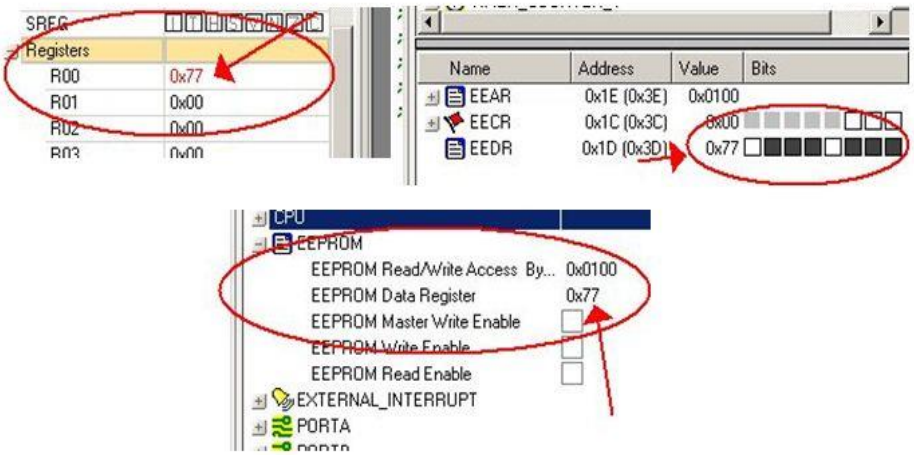

Рис 2. Перевірка процедури запису комірки $100 пам’яті EEPROM

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ УПРАВЛЕНИЯ ПОИСКОМ НА ПРИМЕРЕ ИНТЕЛЛЕКТУАЛЬНОЙ ИГРЫ

Копёнкин В.С. ст. гр. КИ-15бд ст. преподаватель Холодняк В.Н.
Восточноукраинский национальный университет им. В.Даля

Практически все игры можно представить в виде так называемых деревьев решений, где каждый узел будет представлять собой один шаг решения задачи (ход в игре), ветвь в дереве соответствует решению, которое ведёт к более полному решению, листы представляют собой окончательное решение (итоговые позиции). Наша цель – найти в дереве лучший путь от корня до листа.

Деревья решений обычно невероятно велики. Например, для игры в крестики-нолики дерево содержит более полумиллиона узлов. Понятно, что в шашках, шахматах, деревья на порядок больше. Но, тем не менее, на примере этих игр можно рассмотреть методы, позволяющие находить оптимальные решения в очень больших деревьях.

Типовая СППР (система поддержки принятия решений), предназначенная для поиска максимально выгодного «хода» в интеллектуальной игре, представляет собой объединенную информационным процессом совокупность технических средств и программного обеспечения, работающих во взаимосвязи с человеком (коллективом людей), и способную на основе сведений и знаний при наличии мотивации синтезировать алгоритм действий.

В интеллектуальных играх соревнование между участниками заключается в том, что они поочередно принимают решения, не зная, каким будет следующее решение противника. Классический подход, реализуемый мыслящим существом для решения этой задачи, состоит в прогнозировании последующих ходов – своих и ответных ходов противника. Таким образом, может быть построено дерево (или граф) допустимых ходов и возможных игровых позиций.

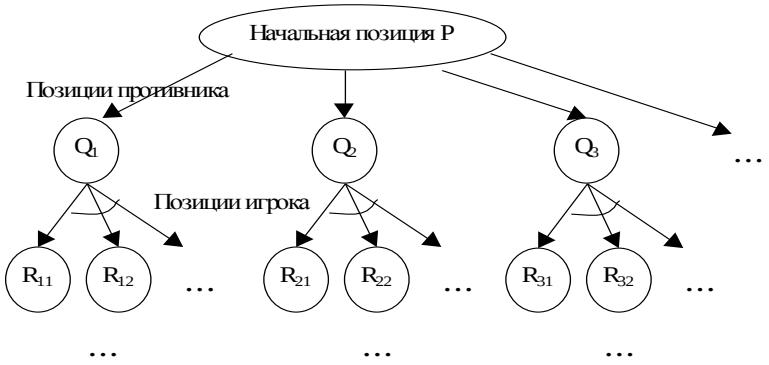

Рисунок 1 - Дерево (или граф) допустимых ходов и возможных игровых позиций

Если необходимо найти какое-нибудь решение задачи, то организуется простейший поиск в «И – ИЛИ» дереве, который заключается в систематическом и полном просмотре дерева, не руководствуясь при этом какими – либо эвристиками. Для сложных задач подобные процедуры неэффективны из–за большой комбинаторной сложности пространства поиска. Например, для игры в шахматы пространство поиска оценивается в 10120 позиций! Поэтому необходимы эвристические алгоритмы управления поиском. Основная идея этих алгоритмов – просмотр только части дерева ходов.

Существует два основных алгоритма поиска «лучших ходов»: минимаксный и альфа-бета алгоритмы.

Это стандартные методы поиска, используемые в игровых программах. В минимаксном алгоритме дерево ходов просматривается по одной из ветвей до максимальной глубины (обычно несколько ходов) и оценивается позиция. Очень многое зависит от оценочной функции, которая для большинства игр, является приближенной эвристической оценкой шансов на выигрыш одного из участников игры.

Пользуясь этим минимаксным принципом и зная значения оценок для всех вершин подножия дерева поиска, можно определить оценки всех вышерасположенных вершин дерева. Оценки вершин подножия дерева делаются с помощью некоторой оценочной функции и называются статическими. Затем, двигаясь по дереву снизу – вверх, определяется оценка (динамическая) внутренних вершин.

Минимаксный алгоритм оценки может быть более экономным. Тут на помощь приходит Альфа-бета алгоритм. Для экономии памяти может быть использована следующая идея. Предположим, что есть два варианта хода. Как только стало известно, что один ход явно хуже другого, то можно принять правильное решение, не выясняя, насколько в точности он хуже. Этот принцип может быть использован для сокращения дерева поиска.

Ключевая идея альфа – бета отсечения состоит в том, чтобы найти ход не обязательно лучший, но «достаточно хороший» для того, чтобы принять правильное решение. Эту идею можно формализовать, введя два граничных значения, обычно обозначаемых Альфа и Бета, между которыми должна находиться рабочая оценка позиции.

Реализация Альфа-бета алгоритма:

```
int AlphaBeta (pos, depth, alpha, beta)
{   if (depth == 0) return Evaluate(pos);
    best = -INFINITY;
    succ = Successors(pos);
    while (not Empty(succ) && best < beta)
    {   pos = RemoveOne(succ);
        if (best > alpha) alpha = best;
```

```
        value = -AlphaBeta(pos, depth-1, -beta, -alpha);
        if (value > best) best = value;
      }
      return best;
    }
```

Преимущество заключается в более раннем выходе из цикла while; значение best, которое равняется или превышает beta-грань, называется отсечением (cutoff). Эти отсечения полностью безопасны (корректны), потому что они гарантируют, что отсекаема часть дерева хуже, чем основной вариант. Самый большой выигрыш будет достигнут, когда на каждом уровне дерева лучшая последующая позиция будет рассмотрена сначала, т.к. эта позиция будет частью основного варианта (который мы хотим обнаружить как можно раньше) или это заставит отсечению произойти как можно раньше.

При оптимальных обстоятельствах перебор с альфа-бета отсечением должен просмотреть W^((D+1)/2) + W^(D/2) - 1 позицию. В то время как число позиций, которое должно быть просмотрено минимаксным алгоритмом - W^D, где W - ширина дерева (среднее количество ходов, возможных в каждой позиции) и D - глубина дерева, что свидетельствует о чрезвычайной неэффективности этого метода. В свою очередь, альфа-бета отсечение позволяет достигать примерно вдвое большей глубины за то же самое время. Большее количество позиций будет просмотрено в том случае, если при переборе не совершается упорядочение ходов.

ЛИТЕРАТУРА:

1. Глушань В.М., Карелин В.П., Кузьменко О.Л. Нечеткие модели и методы многокритериального вьйора в интеллектуальных системах поддержки принятия решений // Известия ЮФУ. Технические науки. Тематич. выпуск «Интеллектуальные САПР». 2009. №4.

ВИКОРИСТАННЯ ПАТЕРНУ ДЛЯ ПОБУДОВИ GPSS СЕРЕДОВИЩА ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ НА ОСНОВІ ПАТЕРНУ ІНЖЕКЦІЇ ЗАЛЕЖНОСТЕЙ

Бритик О. С. АТП -12Д, Борисова М.В. ФЛ-631
Сілютіна І.М., доц. кафедри філософії культури і культурології, к.п.н.
Східноукраїнський національний університет ім. В. Даля

Система GPSS була запропонована задовго до створення технології об’єктно-орієнтованого програмування (ООП). Використання принципів ООП для створення GPSS подібної системи імітаційного моделювання обіцяє дуже великі перспективи. Для побудови сучасних об’єктно-орієнтованих програмних комплексів широко використовується підхід, що базується на основі застосування патернів програмування (Pattern). Патерн - це високоякісне рішення однієї з стандартних задач у програмуванні із застосуванням ООП.

Для забезпечення розширення набору блоків розроблена система GPSS-Fortran [2], яка допускає побудову користувальницьких блоків на універсальній мові Fortran. Алгоритмічна мова Fortran була розроблена у 50-х роках минулого сторіччя і в наш час не дуже поширена із-за властивих недоліків.

Багато робіт присвячено розробці патернів програмування [4]. Але відсутні роботи, щодо застосування патернів для розробки GPSS-подібної системи імітаційного моделювання.

Метою публікації є дослідження системи дискретного імітаційного моделювання на базі сучасних Java технологій [3]. Вибір цієї платформи дозволяє використовувати систему моделювання у режимі хмарних обчислень. У такому разі у корпоративній або глобальній